

Continual Learning with Hebbian Plasticity in Sparse and Predictive Coding Networks: A Survey and Perspective

Ali Safa

College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar

E-mail: asafa@hbku.edu.qa

November 2024

Abstract. Recently, the use of bio-inspired learning techniques such as Hebbian learning and its closely-related Spike-Timing-Dependent Plasticity (STDP) variant have drawn significant attention for the design of compute-efficient AI systems that can continuously learn on-line at the edge. A key differentiating factor regarding this emerging class of neuromorphic continual learning system lies in the fact that learning must be carried using a data stream received in its natural order, as opposed to conventional gradient-based offline training, where a static training dataset is assumed available a priori and randomly shuffled to make the training set independent and identically distributed (i.i.d). In contrast, the emerging class of neuromorphic continual learning systems covered in this survey must learn to integrate new information on the fly in a non-i.i.d manner, which makes these systems subject to catastrophic forgetting. In order to build the next generation of neuromorphic AI systems that can continuously learn at the edge, a growing number of research groups are studying the use of Sparse and Predictive Coding-based Hebbian neural network architectures and the related Spiking Neural Networks (SNNs) equipped with STDP learning. However, since this research field is still emerging, there is a need for providing a holistic view of the different approaches proposed in the literature so far. To this end, this survey covers a number of recent works in the field of neuromorphic continual learning based on state-of-the-art Sparse and Predictive Coding technology; provides background theory to help interested researchers quickly learn the key concepts; and discusses important future research questions in light of the different works covered in this paper. It is hoped that this survey will contribute towards future research in the field of neuromorphic continual learning.

Supplementary Material

Please find the code for a demonstration of Continual Learning with Sparse Coding Networks (presented in Section 2.2) at <https://tinyurl.com/y8n5a3tf>

1. Introduction

The remarkable ability of biological systems to learn and adapt in real-time to their environment while consuming little power compared to conventional Deep Learning systems has been a major motivation behind the study of neuromorphic computing [1, 2, 3, 4]. Indeed, biological entities significantly differ from the current Deep Learning AI systems in terms of their *learning paradigm* [5]. While the large majority of today's AI systems rely on the availability of training datasets *captured beforehand*, biological entities are capable to learn on the fly using the perception data (e.g., vision) received in a real-time order [7]. Hence, biological entities seem to be capable of learning in non-stationary data streams where data points cannot be assumed independent and identically distributed (non-i.i.d) [8, 9]. This learning paradigm is referred to as *Continual Learning* in literature [8]. This comes in great contrast with the gradient-based *offline* training setting, where the offline training data is randomly shuffled in order to make the data source i.i.d, which is needed for stochastic gradient descent to converge successfully [8].

In recent years, the remarkable ability of biological entities for continual learning has triggered research in the emerging field of *neuromorphic* continual learning, which seeks to study how bio-inspired neural network architectures such as non-spiking Artificial Neural Networks (ANNs) equipped with Hebbian learning and Spiking Neural Networks (SNNs) equipped with Spike-Timing-Dependent Plasticity (STDP) can be used for designing continual learning systems, without suffering from the compute- and memory-intensive overheads of traditional Deep Learning systems [10, 11, 12, 13, 14, 15, 16]. Following these realisations, a number of works covering a wide range of applications have been proposed, from task- and class-incremental learning setups [10] to the continual learning of object detection and robot navigation [7, 9].

Studying the conception of neuromorphic continual learning system is an important emerging topic as it has the potential to enable a new class of extreme edge devices that can learn while performing inference at the edge [17]. This novel mode of operation is significantly different to the current way that most neuromorphic systems operate with weights learned during an *offline* training phase that do not change or adapt during the inference phase [4]. Hence, the study of neuromorphic continual learning will lead to novel functionalities and features such as:

- The *personalization* of extreme edge devices to their specific environment or user, which will lead to better algorithmic performance [9].
- Alleviating privacy issues: these new algorithms will be able to refine the weights of their neural networks without resorting to compute-expensive offline training phases, which require a transfer of data from the extreme edge device to the cloud [17].
- On-chip learning: understanding how to design learning algorithms that can learn on a stream of real-world data shown in its real-time order considerably reduces the memory requirements compared to conventional shuffled datasets used in traditional

offline training. This reduction in memory requirements would in turn reduce the chip area and power consumption of the overall integrated circuits [6].

But, to the best of our knowledge, a holistic survey giving an ensemble view of the different state-of-the-art works carried in the field of *neuromorphic continual learning* has not yet been provided, surely due to the more recent nature of this field compared to other sub-fields of neuromorphic computing which have been researched for a longer time. Hence, inspired by a number of prior surveys [1, 2, 3, 4] covering neuromorphic topics such as local synaptic plasticity [5] and event-based computation [17], this paper provides a holistic survey of recent works in the field of neuromorphic continual learning, using state-of-the-art Sparse and Predictive Coding architectures which have gained huge interest in this field. Compared to the closely-related survey proposed by R. Mishra *et al.* [6], this paper provides a *complementary* view of works in the field of neuromorphic continual learning, where focus is mostly made on describing *system-level* works reported in literature, complementary to the focus on *specific techniques* mostly adopted in the survey of R. Mishra *et al.* [6].

In this survey, both Hebbian learning in non-spiking ANNs and its *closely-related* STDP variant in SNNs are covered, since it is widely accepted that STDP can be seen as a specific implementation of Hebbian learning in the spiking domain [22], which motivates our covering of both related notions in this paper. Since many of the related works are based on the theoretical concepts of Sparse Coding and Predictive Coding [18, 23], this paper also provides useful background theory on the key concepts behind the neuromorphic continual learning systems reported in literature. Finally, this survey discusses the future perspectives and suggests a number of future research questions for further advancing the field of neuromorphic continual learning.

This paper is organized as follows. Preliminaries on the relation between Continual Learning and Sparse & Predictive Coding Networks are provided in Section 2. Background theory is provided in Section 3. An overview of notable works in continual learning with non-spiking Hebbian plasticity is covered in Section 4. Transitioning to the spiking domain, an overview of notable works in continual learning with SNNs and STDP plasticity is given in Section 5. Discussions and future perspective on bio-plausible continual learning techniques are provided in Section 6. Finally, conclusions are provided in Section 7.

2. Preliminaries on the relation between Continual Learning and Sparse & Predictive Coding Networks

2.1. Motivations

The approach adopted in this survey seeks to cover the use of Sparse & Predictive Coding Neural Networks for Continual Learning applications, as state-of-the-art neural network design approaches that closely incorporate bio-plausible Hebbian learning [26]. Following background literature, we use the terminology of *Predictive Coding* to refer

to a multi-layer cascade of single Sparse Coding layers [18, 23].

It is important to state that this survey will distinguish the covered works between the non-spiking and spiking cases. In the non-spiking case, the neural architectures correspond to a connectionist implementation of Sparse or Predictive Coding algorithms. In the spiking case on the other hand, the neural architectures make use of spiking neurons and represent a Spiking Neural Network (SNN) implementation of Sparse or Predictive Coding algorithms. In addition, the SNN cases make use of the Spike-Timing-Dependent Plasticity (STDP) rule [22], as a spiking implementation of Hebbian learning. We make this distinction between the non-spiking and spiking case to be in-line with the hardware-efficient SNN-STDP processors that have been remarkably designed by the Neuromorphic community [2, 68].

2.1.1. On the link between Sparse & Predictive Coding networks and Hebbian learning

Sparse and Predictive Coding architectures are known to be tightly linked with the concept of Hebbian learning, since it can be mathematically shown that the learning that takes place within Sparse or Predictive Coding systems when implemented as neural networks boils down to local Hebbian learning mechanisms [26, 73] (see Equation 5 in Section 3.1.1 of this survey). Today, a growing research community is interested in studying Hebbian learning networks via the use of Sparse and Predictive Coding architectures, since this setting provides a clear mathematical model for understanding what precise objective function the Hebbian plasticity mechanisms are optimizing [6, 11, 13, 26, 73]. This is in contrast to the use of prior-art Hebbian network architectures such as Reservoir-style and Winner-Take-All (WTA) networks where the energy function minimized by the Hebbian network is not clearly formulated, leading to un-modeled and less-predictive behaviors [23]. On the other hand, this key observation made by a number of researchers in the Neuromorphic community about the tight link between the concise mathematical formalism of Sparse or Predictive Coding theory and Hebbian learning has enabled the inception of more powerful Hebbian learning networks, enabling the tackling of new challenging tasks such as drone navigation, people detection, agent control and so on [7, 12, 13].

2.1.2. On the link between Continual Learning and Sparse & Predictive Coding networks

Sparse and Predictive Coding networks (which implement local Hebbian-style plasticity) are increasingly being studied under a Continual Learning setting because they are believed to be key enablers of future Neuromorphic devices that could personalize themselves to their working environment and learn on the fly in an energy-efficient way (like biological entities, but unlike current offline-trained deep learning systems). In Deep Learning approaches, Continual Learning is often achieved by computing complex regularization terms involving second-order Hessian matrix computations and their related variants [43]. But another subcategory of Continual Learning techniques within Deep Learning are referred to as *Parameter Isolation* methods [8]. As their name suggest, Parameter Isolation methods seek to perform continual learning by allocating

(in a hard or soft manner) portions of the total neural network resources to each different latent task domain that the network is seeking to learn. This corresponds to what Sparse and Predictive Coding systems can achieve due to their sparsity constraints and re-projection error minimization during learning. Indeed, the sparsity constraints in their neural activations leads to a soft compartmentalizing and isolation of their weights, since output neurons that are not firing do not provoke a change in their associated synapse strength, by virtue of Hebbian Learning. On top of this, the re-projection error minimization also helps with the compartmentalizing of the learned information since it encourages the weights to learn complementary visual fields (i.e., a diverse set of matched filters) where each visual field is better tailored to a particular input data context within the stream of non-identically-distributed data used during the continual learning process. In Section 2.2 we provide an illustrative experiment clearly showing the usefulness of Sparse and Predictive Coding networks for Continual Learning.

2.1.3. On the importance of Sparse & Predictive Coding networks, Hebbian learning and Continual Learning Finally, we would like to briefly motivate and explain why studying the application of Sparse and Predictive Coding networks (which implement Hebbian plasticity) for Continual Learning is important from both a computational Neuroscience and Engineering point of view.

First, from a computational Neuroscience point of view, it is well known that backpropagation-style learning algorithms are, in the overwhelming majority of works, not considered biologically plausible due to their non-local nature [75]. Now, when it comes to Continual Learning, using regularization-based techniques that compute variants of Hessian matrices for the network weights [43] make this even more implausible from a biological point of view. On the other hand, it is well-known that Hebbian learning has strong biological plausibility [31], and the same can be said about Sparse and Predictive Coding Networks which have been proposed in a number of past seminal works as plausible architecture for e.g., the V1 region behavior in the brain [19]. In addition to this, it is well-known that biological agents have a remarkable capability at adapting their behavior and *learning in real time*, as in the Continual Learning case. Hence, this leads to a key research question: how can we design Continual Learning systems using more bio-plausible architectures such as Sparse and Predictive Coding networks (which learn through local Hebbian plasticity).

Second, from an engineering point of view, it is believed by the community that being able to build energy-efficient continual learning systems that can learn and personalize themselves on the fly would constitute a major realization in the Neuromorphic Computing field, compared to the ever dominance of offline training algorithms in today’s deep learning systems [6]. Hence, performing research in this field has a huge potential since it could potentially enable the creation of a whole range of novel IoT and edge consumer devices that could intimately adapt themselves to their user or environment, so as to attain higher performances, without resorting to costly offline training and its privacy preservation concerns. This constitutes a

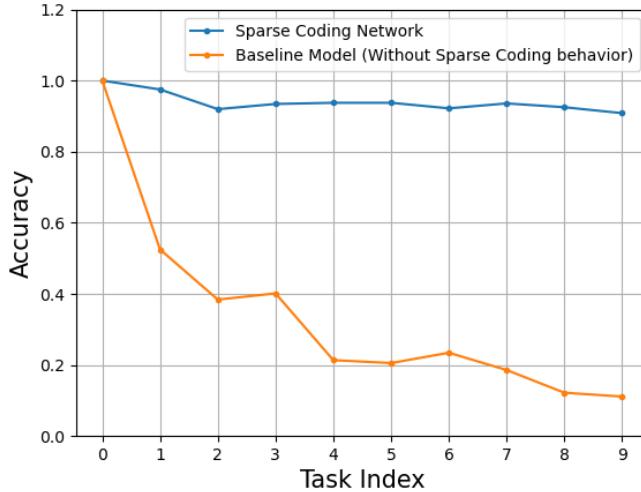


Figure 1. *Illustration of the usefulness of Sparse Coding Networks for Continual Learning applications.* Compared to the baseline model (supervised linear 1-layer network), the 1-layer Sparse Coding network outperforms the baseline model in terms of classification accuracy retention, as new digit classes are fed to the networks during the continual learning process.

major motivation behind the study of Hebbian-driven Sparse and Predictive Coding Networks for continual learning, as an emerging, state-of-the-art approach for designing local learning algorithms. These local Hebbian learning networks could then be fit for deployment in Hardware-efficient Neuromorphic chips developed by the vibrant Neuromorphic hardware community [2, 68].

2.2. Experimental motivation on the usefulness of Sparse & Predictive Coding networks for Continual Learning

In addition to the preliminary motivations introduced in Section 2.1 on the tight links between Sparse & Predictive Coding networks, Hebbian learning and Continual Learning, this section provides an introductory experimental illustration of *the usefulness of Sparse & Predictive Coding networks for tackling Continual Learning scenarios*.

In this continual learning experiment, we compare the classification accuracy of a 1-layer Sparse Coding network to the accuracy of a fully-connected 1-layer baseline model (i.e., without Sparse Coding behavior). We implement both networks in `python` using the `pytorch` library [76]. Classification is done using population coding [74]. Our continual learning scenario is set up as follows. Images from the `digits` dataset of the `sklearn` library [77] are fed in a class-incremental manner to each neural network model. This means that all images corresponding to the class "0" are first used for learning; then, images of the class "0" are not shown anymore and only images from the class "1" are shown. The continual learning process continues until all the 10 classes

are covered. This continual learning setup is naturally challenging since it makes the learning prone to *catastrophic forgetting*. Indeed, when feeding the images of class "1", the models could be prone to over-fitting on the images of class "1" and in turn, forget what was learned with the images of class "0" etc.

Fig. 1 shows the result of this illustrative experiment, where it can be clearly remarked that the Sparse Coding network significantly outperforms the baseline model in terms of accuracy retention as new classes are fed to the networks. Hence, this experiment clearly motivates the use of Sparse & Predictive Coding networks for continual learning applications and explains why a growing number of research teams are now exploring the use of Sparse Coding-type networks for building the next generation of Neuromorphic continual learning systems.

Finally, the source code for the experiment in Fig. 1 is released as open-source at <https://tinyurl.com/y8n5a3tf>

3. Background theory

This Section provides useful background information on the main neural network components used in the various works that are surveyed in this paper. First in the *non-spiking* domain, since many of the neural network designs surveyed in this paper are based on the Sparse Coding (SC) [19] and related Predictive Coding architectures (PC) [20, 21], Section 3.1 overviews the working principle behind SC and PC, as well as providing a view of how Hebbian learning techniques are used within the SC and PC contexts. Then, transitioning to the *spiking* case, Section 3.2 discusses how SNN architectures based on the SC principle are built in literature, and provides a view of how STDP learning techniques are used within the SNN context. Finally, Section 3.3 introduces the Continual Learning (CL) problem and its effect on learning performance.

3.1. Sparse Coding and Predictive Coding Neural Architectures with Hebbian Learning

3.1.1. Sparse Coding with ANNs and Hebbian plasticity

An important number of state-of-the-art works in literature model the neural activity of biological agents through Sparse Coding (SC) and Predictive Coding (PC) neural networks [18], [23]. This is motivated by the fact that SC and PC have been shown to be bio-plausible models of cortical neural ensembles, as observed in the mushroom body of an insect's brain [24]. In a probabilistic sense, SC models the output and weight distribution of ANNs equipped with Hebbian learning through an identically-distributed Gaussian likelihood model with a Laplacian prior on the neural activity c :

$$p(c|o, \Phi) \sim \exp(-\|\Phi c - o\|_2^2) \exp(-\lambda \|c\|_1) \quad (1)$$

where o is the input of dimension N , c is the output of dimension M , Φ is the $N \times M$ weight matrix of the layer (also called *dictionary*), and λ is a hyper-parameter setting

the scale of the Laplacian prior. Using a Laplacian prior is motivated by the fact that its distribution shape promotes sparsity in the output neural activation, in an analogous way to how bio-plausible SNNs using e.g., Leaky Integrate-and-Fire (LIF) neurons promote sparsity in their output codes [23].

In SC (1), the inference of the output activation c and the learning of the network weights Φ can be obtained through [23]:

$$C, \Phi = \arg \min_{C, \Phi} \sum_l \|\Phi c_l - o_l\|_2^2 + \lambda \|c_l\|_1 \text{ with } C = \{c_l, \forall l\} \quad (2)$$

Equation 2 is typically solved via Proximal Stochastic Gradient Descent [25], by *alternating* between: *a*) the inference of c_l , given the current input o_l and the weight Φ and *b*) the learning of Φ , given the current c_l and o_l .

Neural network layers solving the SC problem (2) can be implemented following the dynamical system given in (3), where T denotes the transpose, η_c is the coding rate, η_d is the learning rate and $\mathbf{Prox}_{\lambda \|\cdot\|_1}$ is the proximal operator to the l_1 norm (non-linearity) [26]. For each input o_i , the neural and weight dynamics of the Hebbian network follows the update rules in (3) for an arbitrary number of iterations N_{it} , in order to infer the corresponding c_l and learn Φ [23].

$$\begin{cases} c_l \leftarrow \mathbf{Prox}_{\lambda \|\cdot\|_1} \{c_l - \eta_c \Phi^T (\Phi c_l - o_l)\} \\ \Phi \leftarrow \Phi - \eta_d (\Phi c_l - o_l) c_l^T \end{cases} \quad (3)$$

with $\mathbf{Prox}_{\lambda \|\cdot\|_1}$ acting as the neural non-linearity:

$$\mathbf{Prox}_{\lambda \|\cdot\|_1}(c_i) = \text{sign}(c_i) \max(0, |c_i| - \eta_c \lambda), \forall i \quad (4)$$

Algorithm 1 summarizes the alternating-descent procedure classically used to solve (2) [23]. This procedure will also be used by the neural network architecture shown in Fig. 2, essentially executing Algorithm 1 through its explicit neural wiring and Hebbian plasticity.

From a neural network topology point of view, the dynamical system of (3) can be implemented as the network architecture in Fig. 2, composed of an *output* layer using the non-linearity (4) and a *linear* layer estimating the error vector \bar{e} , where all weight updates follow the form of the standard Hebbian rule [23]:

$$w_{ij} \leftarrow w_{ij} - \eta_d y_i x_j \quad (5)$$

In (5), η_d is the learning rate, y_i is the post-synaptic (or output) activation value of neuron i and x_j is the pre-synaptic (or input) value received by the j^{th} synapse of neuron i . The error vector e outputted by the linear layer in Fig. 2 encodes the re-projection error in (2) as:

$$e = \Phi c - o \quad (6)$$

In Fig. 2, Hebbian learning (5) is applied to the weight matrix Φ of the linear layer following:

$$\Phi_{ij} \leftarrow \Phi_{ij} - \eta_d e_i c_j \quad (7)$$

Algorithm 1 joint Dictionary Learning and Basis Pursuit

Input: $\bar{s}_k \forall k$: input vector stream, η_c : coding rate, η_d : learning rate, λ_1 : regularization, N_{it} : number of iterations.

- 1: **Init.:** $\Phi_{ij} \leftarrow \mathcal{N}(0, \sigma_w)$ (zero-mean normal distribution with std. deviation $\sigma_w \sim 0.01$), $\bar{c} \leftarrow 0$
 - 2: **for** $l \in \{1, \dots, T_{end}\}$ (feed data sequence) **do**
 - 3: **for** $p = 1, \dots, N_{it}$ (local coding iterations) **do**
 - 4: $\bar{c}_l \leftarrow \text{Prox}_{\eta_c \lambda_1 \|\cdot\|_1} \{\bar{c}_l - \eta_c \Phi^T (\Phi \bar{c}_l - \bar{s}_l)\}$ // see (3)
 - 5: **end for**
 - 6: **for** $p = 1, \dots, N_{it}$ (local learning iterations) **do**
 - 7: $\Phi \leftarrow \Phi - \eta_d (\Phi \bar{c}_l - \bar{s}_l) \bar{c}_l^T$
 - 8: **end for**
 - 9: **end for**
-

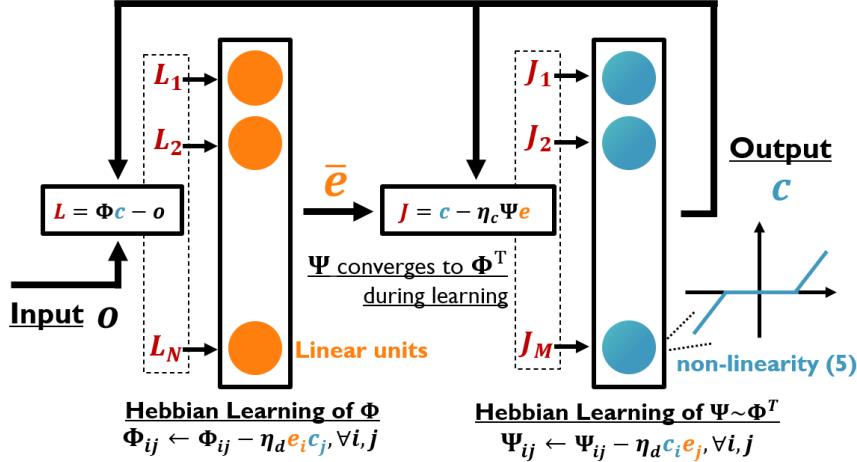


Figure 2. *Baseline Hebbian network architecture used in this work. The dynamics of the network follow (3) and minimize (2), given subsequent input vectors o . Each layer possesses its own weight matrix Φ, Ψ which evolve through Hebbian plasticity ($\Psi \sim \Phi^T$ in (3), as an independent, local set of weights).*

where pre-synaptic c_j is the j^{th} element of the output layer and post-synaptic e_i is the i^{th} element of the linear layer with output vector e (6).

Similarly, Hebbian learning is locally applied to the weight matrix Ψ of the output layer:

$$\Psi_{ij} \leftarrow \Psi_{ij} - \eta_d c_i e_j \quad (8)$$

making Ψ converge to the *transposed* of the dictionary weight matrix Φ during the learning process $\Psi \sim \Phi^T$ (since the pre-synaptic and post-synaptic signals are swapped between Ψ and Φ) [23].

3.1.2. Predictive Coding with ANNs and Hebbian plasticity

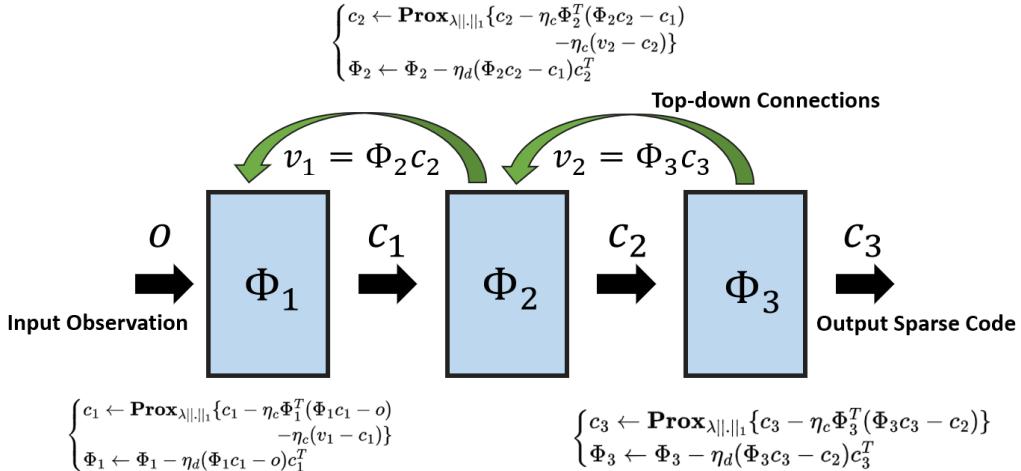


Figure 3. Predictive Coding neural network architecture. In this example, a 3-layer PC network is shown, where each blue box represents a SC network instantiation of the Hebbian network shown in Fig. 2. The arrows indicate the top-down connections from each upstream layer to its previous one.

Closely related to the SC problem, Predictive Coding (PC) can be seen as a multi-layer SC system where each layer r not only minimizes its own re-projection error $\|\Phi_r c_r - o_r\|_2^2$ between its output sparse code c_r and input observation o_r , but also seeks to minimize the re-projection error $\|\Phi_{r+1} c_{r+1} - o_{r+1}\|_2^2$ of the next downstream SC layer $r + 1$, with $o_{r+1} = c_r$. Hence, each SC layer is also trying to *predict* the error done by the layer coming after it, in order to minimize the global loss function of the complete system, corresponding to the sum of all re-projection errors [27]:

$$\Phi_{1,\dots,R}, c_{1,\dots,R} = \arg \min_{\Phi_{1,\dots,R}, c_{1,\dots,R}} \sum_{r=1}^R \|\Phi_r c_r - c_{r-1}\|_2^2 + \lambda \|c_r\|_1 \quad (9)$$

where R denotes the number of cascaded SC layers and $c_0 = o$ is the input observation data. The PC system can be implemented as a cascade of SC layers (see Fig. 2), where each layer r also receives *top-down* connection $v_r = \Phi_{r+1} c_{r+1}$ from the output sparse code c_{r+1} of the next layer $r + 1$, through its weight matrix Φ_{r+1} . Thanks to the availability of this top-down connection v_r , each SC layer r can also correct for the re-projection error of the next layer $r + 1$ by upgrading the iterative process (3) to the following [27]:

$$\begin{cases} c_r \leftarrow \text{Prox}_{\lambda||\cdot||_1}\{c_r - \eta_c \Phi_r^T (\Phi_r c_r - c_{r-1}) - \eta_c (v_r - c_r)\} \\ \Phi_r \leftarrow \Phi_r - \eta_d (\Phi_r c_r - c_{r-1}) c_r^T \end{cases} \quad (10)$$

where the new term $(v_r - c_r)$ corresponds to the gradient of $\|\Phi_{r+1} c_{r+1} - c_r\|_2^2$ in function of c_r . Therefore, by cascading an arbitrary number R of SC layers following Fig. 3, and by applying Hebbian learning to the weights as done in Fig. 2, a multi-layer network of SC coding layers can be built, which minimizes the PC objective function in (9).

In Section 4, this paper will provide an overview of recent works studying Continual Learning systems using both SC and PC neural network architectures.

3.2. Spiking Neural Networks and STDP Learning

In contrast to the continuous activation functions used in both DNNs and the SC and PC systems presented in Section 3.1, SNNs make use of bio-plausible *spiking* neurons as non-linearity, often modelled by a Leaky Integrate-and-Fire (LIF) activation [28] (see Fig. 4):

$$\begin{cases} \frac{dV}{dt} = \frac{1}{\tau_m}(J_{in} - V) \\ \sigma = 1 \text{ if } V \geq \mu \text{ else } 0 \\ V(t + dt) = 0 \text{ if } V \geq \mu \end{cases} \quad (11)$$

with J_{in} the input current to the neuron (12), σ the spiking output, V the membrane potential, τ_m the time constant governing the membrane potential decay, dt the simulation time step, and μ the neuron threshold [29]. The scalar input current J_{in} is continuously integrated in V following (11). When V crosses the firing threshold μ , the membrane potential is reset back to zero and an output spike is emitted. The input current J_{in} is obtained by filtering the inner product of the neural weights and the spiking inputs through a *post-synaptic current* (PSC) kernel [29] (estimating the *instantaneous* spiking rate):

$$J_{in} = \mathcal{PSC}\{\bar{\phi}^T \bar{s}_{in}(t)\} \quad (12)$$

with $\bar{s}_{in}(t)$ the input spiking vector (originating from e.g. an event camera or other spiking neurons), $\bar{\phi}$ the weight vector and:

$$\mathcal{PSC}\{x(t)\} = x(t) * \frac{1}{\tau_s} e^{-t/\tau_s} \quad (13)$$

denoting the effect of PSC filtering with time constant τ_s . Fig. 4 conceptually illustrates the LIF neuron behavior where the single LIF neuron is connected to a pre-synaptic spiking input s_{in} .

Using the LIF neuron model described above, dynamical neural networks working along the time domain can be designed using a similar connectionist approach as in conventional DNNs and their weights can be learned in order to perform useful tasks.

Often used alongside SNNs, the study of Spike-Timing-Dependent Plasticity (STDP) is actively being pursued by the neuromorphic community as a biologically plausible and spiking local *Hebbian* learning rule [5], avoiding the high memory-access-expensive problem of backprop [30]. STDP *locally* modifies the weights of each neuron following the difference τ_{ij} between the post- and pre-synaptic spike times [31]:

$$W_{ij} \leftarrow \begin{cases} W_{ij} + A_+ e^{-\tau_{ij}/\tau_+}, & \text{if } \tau_{ij} \geq 0 \\ W_{ij} - A_- e^{\tau_{ij}/\tau_-}, & \text{if } \tau_{ij} < 0 \end{cases} \quad (14)$$

with A_+, A_- the potentiation and depression weights, τ_+, τ_- the potentiation and depression decay constants, W_{ij} the j^{th} element of the i^{th} neuron weight vector \bar{W} ,

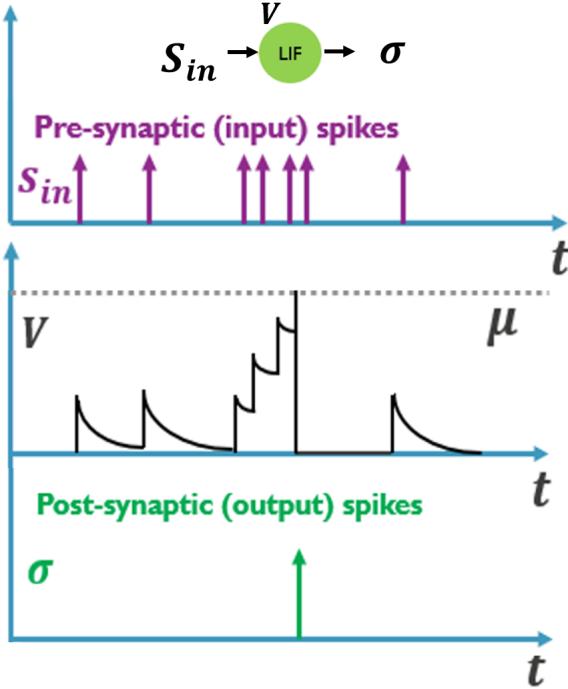


Figure 4. Conceptual illustration of the operation of the LIF neuron. In this simplified example, the LIF neuron is connected to a single spiking source s_{in} via a unit weight.

and τ_{ij} the time difference between the post- and the pre-synaptic spike times across the j^{th} synapse of neuron i (see Fig. 5) [31].

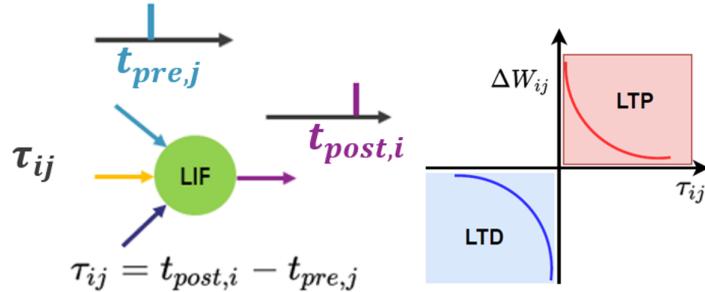


Figure 5. Conceptual illustration of STDP learning. The weight W_{ij} is modified according to the double exponential STDP rule (14) with Long Term Depression (LTD) and Potentiation (LTP) regions.

We further note as $\kappa(\tau)$ the effect of the double exponential STDP kernel on the weight modification in (14):

$$\kappa(\tau) = \begin{cases} A_+ e^{-\tau/\tau_+}, & \text{if } \tau \geq 0 \\ -A_- e^{\tau/\tau_-}, & \text{if } \tau < 0 \end{cases} \quad (15)$$

The STDP rule (14) is closely related to the Hebbian learning rule (5) covered in Section 3.1. Indeed, given a pre- and a post-synaptic spike train $s_{pre}(t)$, $s_{post}(t)$, the weight

modification of the synapse at the post-synaptic time instant t can be written as [32, 23]:

$$\Delta w|_t = \eta_2 \int_{-\infty}^{\infty} s_{post}(t) s_{pre}(t - \tau) \kappa(\tau) d\tau \quad (16)$$

Under this form, (16) is not useful since the post- and pre-synaptic spike trains are not known beforehand. Taking the expected value of (16) over t leads to:

$$\mathbb{E}\{\Delta w|_t\} = \eta_2 \int_{-\infty}^{\infty} \mathbb{E}\{s_{post}(t) s_{pre}(t - \tau)\} \kappa(\tau) d\tau \quad (17)$$

The right-hand side of (17) can be re-written as:

$$\mathbb{E}\{\Delta w|_t\} = \eta_2 \int_{-\infty}^{\infty} \{r_{post} r_{pre} + \mathcal{C}(\tau)\} \kappa(\tau) d\tau \quad (18)$$

where $\mathcal{C}(\tau)$ is the covariance between the pre- and post-synaptic spike train and r_{post}, r_{pre} are the average post- and pre-synaptic spiking rates. We further rearrange (18) into a rate-only term and a covariance term:

$$\Delta w_{STDP} \equiv \mathbb{E}\{\Delta w|_t\} = \eta_2 (A_+ \tau_+ - A_- \tau_-) r_{post} r_{pre} + \eta_2 \int_{-\infty}^{\infty} \mathcal{C}(\tau) \kappa(\tau) d\tau \quad (19)$$

Furthermore, it can be shown [33] that the covariance term in (19) can be ignored since output spikes are caused by a large number of different input spikes, making the output spike *uncorrelated* to each individual input spikes due to the central limit theorem. This effect has also been experimentally assessed and confirmed in [33].

Therefore, the full STDP mechanism Δw_{STDP} is approximated as follows in the remainder of the *derivations* (even though full STDP is indeed used in the *actual implementation* of the various SNN-STDP networks covered in this survey):

$$\Delta w_{STDP}\{s_{post}, s_{pre}\} \approx \eta_2 (A_+ \tau_+ - A_- \tau_-) r_{post} r_{pre} \quad (20)$$

where it is assumed (without loss of generality) that:

$$A_+ \tau_+ - A_- \tau_- = 1 \quad (21)$$

in order to not modify the resulting learning speed (setting $A_+ \tau_+ - A_- \tau_- > 0$ is sufficient for the learning effect to take place).

Therefore, since the effect of STDP is analogous to the effect of Hebbian learning in the local spiking *rate* domain (i.e., when the local spike rate is encoding the neural activation value), STDP can be used *in place* of Hebbian plasticity in the neural architecture of Fig. 2. In addition, the non-spiking neurons found in the architecture of Fig. 2 can be replaced by pairs of push-pull *spiking* LIF neurons (see Fig. 6) in order to replicate the effect of the $\text{Prox}_{\lambda \|\cdot\|_1}$ neural non-linearity (4) in the local spiking rate domain. Indeed, it was remarked in [23, 33] that wiring two LIF activations in a push-pull setting approximates the $\text{Prox}_{\lambda \|\cdot\|_1}$ behavior (4) in terms of the average

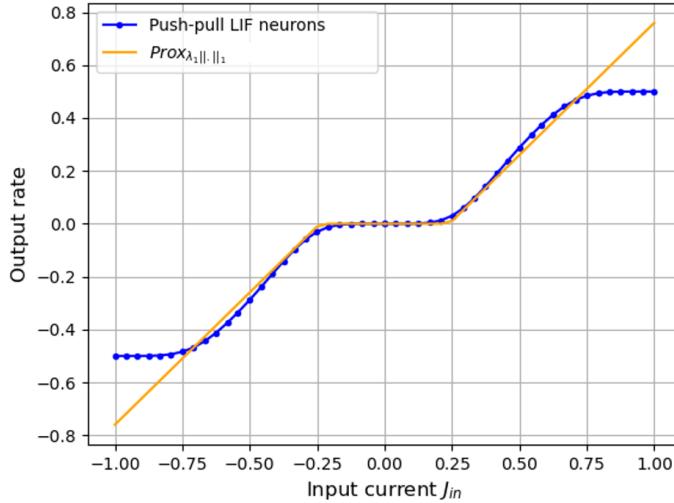


Figure 6. *Input-output relationship of a push-pull LIF neuron pair in the average spike rate domain.* Simulated using a single spiking input connected with a weight of 1 with illustration parameters $\tau_s = 0.01s$, $\mu = 0.15$, $\tau_m = 0.002s$ in (11), (13).

output spike rate in function of the average *input* spike rate, with output saturation for large input rates (i.e., for the spiking probability of a time bin approaching 1) [29]. Fig. 6 shows the simulation result of a pair of push-pull LIF neurons compared to the shape of $\text{Prox}_{\lambda \parallel \cdot \parallel_1}$.

In sum, following the observations provided in this Section about the commonalities between both STDP and Hebbian learning, and between spiking LIF neurons and the $\text{Prox}_{\lambda \parallel \cdot \parallel_1}$ non-linearity used in SC and PC (see Section 3.1), a number of work [34, 13, 9, 7] have used SNNs equipped with STDP (SNN-STDP) in a *Continual Learning* setting wired following the architecture in Fig. 2 (where all neurons are replaced with LIFs and Hebbian learning is replaced with STDP). This enables the extension of SC and PC to a more bio-plausible setting, using an SNN-STDP network instead of the non-spiking architecture in Fig. 2. An overview of recent works in Continual Learning with SNN-STDP will be provided in Section 5.

For the sake of completeness, it is important to mention that other flavors of the STDP learning rule exists, such as reward-modulated STDP [35] or anti-Hebbian STDP [7]. We refer the interested reader to the survey proposed in [5] which gives an overview of local plasticity rules in neuromorphic systems.

3.3. The Continual Learning Problem

In contrast to the conventional *offline* training of neural networks with training sets captured beforehand and *shuffled* before being fed to the networks, continual learning is concerned with the more challenging scenario whereby a neural network must learn while being fed with a continuous stream of data presented in its natural, real-time order (see Fig. 7).

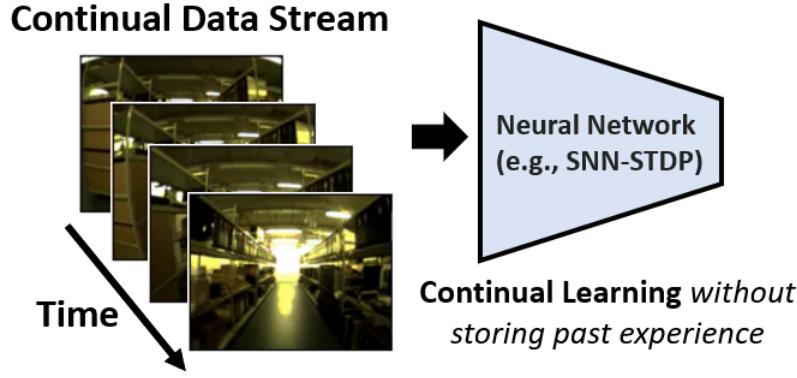


Figure 7. The continual learning scenario where a neural network (e.g., using an SNN-STDP architecture) continuously refine its weights using data points received in their natural, real-time order (without buffering and shuffling past observations in memory). At the same time, the neural network is performing inference at the edge.

To illustrate this, let us consider a toy example where a neural network must learn to distinguish between class A and class B . To achieve this, the typical method for training the network would consist of first *shuffling* the training set containing both classes and then using this *shuffled* dataset for training the network using a mini-batch stochastic gradient descent (SGD) approach. In contrast, let us now consider the continual learning case, where the network must learn while receiving the class A and class B data in a *non-shuffled* order, for example, by feeding all the class A data first, followed by all the class B data during training. It is then clear that learning does not take place efficiently as the network will keep on over-fitting, alternating on class A when all the class A data are shown, followed by an over-fitting on class B when all the class B data are shown, effectively *forgetting* what was learned using the class A images. This phenomenon is famously referred to as the *catastrophic forgetting* problem and constitutes a major challenge for continual learning systems [8].

The non-shuffled data streams found in continual-learning applications are often referred to as *non-stationary* data streams [36] because it cannot be assumed that the prior distribution Γ from which data points are sampled during learning stays the same throughout the learning process. Indeed, when a dataset is shuffled, data points end up being sampled in an independent and identically distributed (i.i.d) manner, as opposed to continuous streams, where subsequent data points \bar{o}_k and \bar{o}_{k+1} are strongly correlated through time k . In the i.i.d case of a shuffled data source, the *expected value* \mathbb{E} of the stochastic gradients used by SGD during DNN learning correctly converges to the gradient of the *full* loss function $\mathcal{L} = \frac{1}{N_d} \sum_{k=1}^{N_d} l(\bar{o}_k)$ computed on all the N_d data points:

$$\mathbb{E}\left[\frac{\partial l(\bar{o}_k)}{\partial W}\right] \rightarrow \frac{\partial \frac{1}{N_d} \sum_{k=1}^{N_d} l(\bar{o}_k)}{\partial W} \quad (22)$$

where W denotes the network weights and l denotes the energy or loss to be minimized (e.g., mean square error, cross-entropy,...).

On the other hand, (22) does not hold anymore in the continual-learning case, where the data points are sampled from a non-stationary source (i.e., a prior distribution Γ that changes during the learning process). Since (22) does not hold, learning is conducted *outside* the assumptions made when using SGD-style techniques and hence learning becomes jeopardized. This discussion does not only hold for SGD, but also for bio-plausible local *greedy descent* learning techniques like Hebbian (5) and STDP learning (14), which also assume data source stationarity (see e.g., [37] for a description of STDP under the light of SGD).

3.3.1. Paradigms and methods for Continual Learning

Continual Learning tasks can broadly be separated in two different categories. The first category which is commonly referred to as *task-incremental* continual learning denotes the scenario where a neural network must learn N_T different tasks $\mathcal{T}_{1,\dots,T}$ (e.g., \mathcal{T}_1 : discriminating between cat and dog images, \mathcal{T}_2 : discriminating between plane and car images, etc.). The learning of these T different tasks is done in an incremental manner, where the neural network is first trained on \mathcal{T}_1 , then trained on \mathcal{T}_2 , and so on. Hence, the goal during learning is to be able to learn new tasks while *not forgetting* the old tasks that were already learned. It is important to note that during the learning of each task *in isolation*, the neural network could have access to a shuffled i.i.d set of data for this particular task [8]. In this case, the non-i.i.d nature of the learning comes from the fact that the data distribution *across* the tasks will exhibit a large variability (e.g., \mathcal{T}_1 containing shuffled images of cats and dogs, and \mathcal{T}_2 containing shuffled images of planes and cars).

Under this *task-incremental* scenario, the boundaries between the different tasks are known. This is in contrast to the second broad category of continual learning paradigm which is referred to as *task-free* continual learning. In the *task-free* scenario, the boundaries between tasks, categories or data sources is unknown during the continual learning process. In this second, more challenging scenario, the continual learning system must learn using the real-time data stream which is usually heavily unbalanced and non-i.i.d. For instance, a notable line of work in the task-free setting consists of having robot agents learning to encode visual sensory data as they explore an unknown environment, for performing tasks such as Simultaneous Localization and Mapping (SLAM) [38, 13, 9].

Regarding methods for alleviating the catastrophic forgetting issues encountered during continual learning, most research has focused on three broad categories of approaches, usually studied in the *less bio-plausible* DNN context. The first category is referred to as *replay methods* and consists of keeping a limited amount of data from each task seen in the past, or by keeping segments of past sensory streams in order to feed them again for training in the future. Hence, this method seeks to alleviate catastrophic forgetting by episodically replaying past training data. Even though useful, this method requires potentially significant memory overheads, with a growing memory size to store all past experiences. This makes it less suited for continual learning in

resource-constrained neuromorphic edge devices where memory consumption is usually limited. A second category of method is referred to as *parameter isolation methods* [8] and seeks to dynamically add new architectural elements to the neural network each time a new task is encountered. Hence, this method is more suited for task-incremental continual learning scenarios. In addition, this method also suffers from dynamically growing memory consumption, making it also less suited for energy- and area-efficient neuromorphic computing in edge devices. Finally, the third broad category of continual learning methods found in DNN research is commonly referred to as *regularization methods*. Regularization methods use weight regularization techniques that first identify the *importance of each weight* for the task that have been learned up to the present point in time. Then, the more a weight is considered important, the less the weight regularization enables that weight to change during the learning of new future tasks. In contrast to the previously mentioned method, the regularization method does not lead to significant memory overheads, and has been demonstrated in both task-incremental and task-free settings [39].

More closely related to ANNs with Hebbian learning and SNNs equipped with STDP, a number of prior work [40] have shown that providing *activation sparsity* in DNNs using lateral inhibition between the neurons of each layer (as in the bio-plausible SC and PC systems covered in Sections 3.1 and 3.2) has a beneficial effect on continual-learning performance, usually leading to a significant boost in the network accuracy after learning. Related to this observation, it has been remarked that gradient sparsity during surrogate gradient-based learning can also have a beneficial effect on continual learning in SNNs [41]. In addition, other works have shown the links between popular regularization techniques used for CL in Deep Learning contexts, with Hebbian learning theory [42].

Hence, the study of bio-plausible neural networks built following the SC and PC approaches using local learning rules such as Hebbian and STDP learning has attracted a growing attention for the design of continual-learning systems for extreme edge applications. To this end, the next Sections of this paper will provide an overview of the recent research done in the design of bio-plausible continual learning systems.

4. Continual learning in Hebbian networks

4.1. Task-incremental Continual Learning

In a recent pioneering work, A. Ororbia *et al.* [10] have proposed a lifelong continual learning system based on Predictive Coding (see Section 3.1.2), and have demonstrated their findings on common *task-incremental* classification problems (see Section 3.3.1 for a description of the task-incremental setting). It is important to note that even though demonstrated on task-incremental benchmarks, the network proposed in [10] can work in a task-free setting, as it has the intrinsic capability of detecting if an incoming sample originates from a new task without receiving external task identifiers. This forms a

particularly innovative aspect of this work.

The network proposed in [10] is built as follows using two distinct neural ensembles. The first neural ensemble is a Predictive Coding network as described in Section 3.1.2. This Predictive Coding network acts as a generative model which can reconstruct its input from its output sparse code (i.e., by re-projection following $o \approx \Phi c$ in (9)). The Predictive Coding network is fed as input a vector $s_{in} = \{o, y, t\}$, combining the input signal o , its one-hot class vector y and a *task-pointer* vector t . The task pointer vector t is produced by a second neural ensemble and helps the predictive coding backbone at better selecting different sub-networks routes for each different task. After the continual learning phase has ended, classification of test data can be done by feeding the input observation only and reconstructing (by re-projection $o \approx \Phi c$) the estimated label \hat{y} .

The task-pointer vector t is produced by a second neural ensemble referred to as the *task selector* network. The motivation behind this design choice comes from neuroscience where it is believed that the basal ganglia plays a role in information routing in the brain, selecting different programs and memory stored in other regions in the cortex. The task selector network detect the beginning of new tasks by monitoring the running re-projection error of the Predictive Coding network in (9). If this running re-projection error exceeds its previous value by at least twice its running standard deviation, a task boundary is detected. This information is used by the task selector to infer a task-pointer vector t from the observation o using a SC Hebbian network.

The authors in [10] benchmark their proposed system against a large number of conventional continual learning methods used in the Deep Learning field, such as Elastic Weight Consolidation (EWC) [43], Synaptic Intelligence (SI) [44], Memory Aware Synapses (MAS) [42] and so on. Experiments are conducted using three common benchmark datasets extensively used in continual learning: *Split MNIST*, *Split NotMNIST* and *Split Fashion MNIST* [10]. These datasets are based on the orginal MNIST, NotMNIST and Fashion MNIST, but are shown in a class-incremental manner, each data class at a time (without revisiting past classes). It is shown in [10] that their proposed method is the top performer in terms of classification accuracy compared to the other competing continual learning methods that were assessed.

4.2. Continual Sequence Modelling

In conventional deep learning systems, sequence modelling is usually carried using Recurrent Neural Networks (RNNs) trained using Back-propagation through time (BPTT). Although useful in the deep learning context, BPTT is a highly compute-expensive, non-local learning rule with little bio-plausibility. This is why an emerging body of work [11, 63] are studying the use of PC networks for continual sequence modelling, without resorting to BPTT as their learning rule [45].

To help uncover roads towards the bio-plausible continual learning of time sequences, L. Annabi *et al.* [11] have proposed an innovative continual sequence learning system based on Predictive Coding. In particular, the authors in [11] take

inspiration from Predictive Coding to build an RNN and study a wide range of different methods for learning the input weights, the recurrent weights and the output weights of their proposed 1-hidden-layer RNN architecture. Their system embarks the predictive coding mechanism for learning the recurrent and input weights of their 1-hidden-layer RNN, while experimenting with a range of non-predictive coding techniques for the output weights (e.g., using EWC [43] for the output weights, among other techniques tried by the authors). Experimental demonstrations are provided using motion-capture trajectories of dimension 62 from the CMU Motion Capture database, denoting the evolution of the joints of a human body in the environment. In addition, the authors also provide experimental demonstrations using the 2D handwritten letter trajectories from the UCI Machine Learning repository [46]. First, the author go on to benchmark different *output weight* learning method and demonstrate that the use of the *Conceptor* method proposed in [47] outperforms both the use of EWC and stochastic gradient descent in terms of sequence prediction error for the learning of the output weights. Then, the learning of the *recurrent weights* is studied, and the authors remarkably show that incorporating learning in the recurrent weights following their predictive coding strategy only slightly improves the sequence prediction error compared to the use of a randomly-initialized RNN with random recurrent weights. Finally, the authors experiments with different *input weight* learning approaches and show that the proposed Predictive Coding approach outperforms the use of stochastic gradient descent. In sum, the author conclude that their work show the usefulness of combining Predictive Coding and the Conceptor method [47] for output weight regularization, since this combination lead to the lowest sequence prediction error in their experiments.

Related to the work in [11], A. Ororbia *et al.* [63] have proposed a novel bio-plausible framework for continual learning in RNNs by locally aligning distributed representations captured by each layer of a standard Predictive Coding architecture, equipped with Hebbian learning. Their method, termed *Parallel Temporal Neural Coding Network* (P-TNCN) is essentially based on the Predictive Coding approach (covered in Section 3.1.2) where the first layer receiving an input o_k at time-step k seeks to predict the input for the next time-step o_{k+1} while producing a hidden state c_k^1 . Then, the next layer receiving c_k^1 seeks to predict the next value of the first hidden state c_{k+1}^1 by producing a second hidden state c_k^2 , and so on for as much layers forming the network. In addition to the Hebbian learning rules already found in the Predictive Coding approach, the authors in [63] augment the Predictive Coding Hebbian mechanisms with an additional Hebbian term (23) which seeks to improve the generalization capability of the network by promoting the learning of a model which also exhibit some level of co-occurrence between the statistics of each layer, while minimizing the predictive coding error as well.

$$\Delta_{\text{HebbReg}} W = - \frac{c_k^l (c_{k-1}^{l-1})^T}{\|c_k^l (c_{k-1}^{l-1})^T\|_2} \quad (23)$$

where l denotes the layer index and k the time step.

Then, the authors in [63] propose an extensive experimental study to demonstrate

the performance of their proposed P-TNCN method by comparing it against two popular methods used for training RNNs in an online fashion (RTRL [48] and UORO [49]). In addition, the proposed system is also compared against the use of *Echo State Networks* (ESN) which are a class of RNNs with untrained recurrent weights and trained output weights. Four different datasets are used during the experiments: *i) Bouncing MNIST*, *ii) Bouncing NotMNIST*, *iii) Bouncing Fashion MNIST* and *iv) Penn Treebank*. In the Bouncing MNIST, each data point is a 20 frames long sequence containing two digits from the standard MNIST dataset randomly bouncing around a 64×64 patch. The Bouncing NotMNIST and Fashion MNIST follow a similar construction, while the Penn Treebank dataset contains sequences of words (sentences) for studying Natural Language Processing (NLP).

Remarkably, the authors in [63] show that their proposed Hebbian-learning P-TNCN system outperforms a number of competing methods by reaching a lower loss on the test set for all bouncing datasets. Their proposed algorithm only falls short compared to BPTT when testing on the Penn Treebank dataset, possibly due to the fact that the text data is highly discontinuous. In addition, the authors show the applicability of their proposed methods for zero-shot adaptation to out-of-sample data, and for the online continual learning of data sequences via generative modelling. In sum, to the best of our knowledge, the work proposed by A. Ororbia *et al.* [63] constitutes one of the most advanced account on the use of ANNs equipped with Hebbian learning for the challenging task of continual sequence modelling.

4.3. Dynamical agent control via Hebbian networks

An emerging number of work have shown the applicability of Hebbian networks for agent control tasks that have been previously solved with Reinforcement Learning techniques. A general trend among these works is the fact that learning can be done with either extremely sparse rewards, or either no reward mechanism at all. This is in contrast to traditional methods used in Deep Reinforcement Learning which usually rely on the availability of dense rewards from the environment. The reason behind this phenomenon lies in the fact that these recently-proposed Hebbian agent control systems are built by taking inspiration from the Active Inference framework [50], which achieves agent control by learning in an *unsupervised* manner a *generative model* of the agent’s environment $P(o, a, s)$, where o is the agent’s observation (e.g., position, camera view, etc.), a is the action performed by the agent and s is the latent state of the agent (i.e., a vector summarizing the state of the agent in the environment). Hence, by learning in an unsupervised manner a probability distribution $P(o, a, s)$ over observations, actions and states, an agent can steer itself towards its goal by determining which sequence of actions $\tilde{a} = \{a_1, \dots, a_L\}$ will maximize the probability P of reaching the goal state s^* , with associated observation o^* [50].

Using Sparse Coding, A. Safa *et al.* [51] have proposed a continual-learning Hebbian Active Inference system and demonstrated its application using the Mountain Car

environment from the Open AI gym suite. Crucially, the authors have shown that their proposed system reaches two orders of magnitude faster convergence rate compared to Q-learning (typically used to solve this task), while not requiring a replay buffer as in traditional reinforcement learning systems. Their proposed Hebbian system architecture is built as follows: a first sparse coding network is used to encode the observation data o (car's position and velocity) and action vector a (car's acceleration) into a latent space s , corresponding to the output sparse code of the Hebbian network in Fig. 2. Then, a second Hebbian network is set up following an auto-regressive approach, which estimates the next future state s_{k+1} given a number of past states $\{s_{k-K}, \dots, s_k\}$ and the corresponding actions that were performed by the agent $\{a_{k-K}, \dots, a_k\}$. By learning this model, the agent can perform *planning* in order to steer itself towards its desired goal (the top of the mountain in the case of the Mountain Car environment). This planning is done by randomly sampling sequences of actions $\tilde{a} = \{a_1, \dots, a_L\}$ and estimating the resulting sequence of states $\tilde{s} = \{s_1, \dots, s_L\}$ using the auto-regressive Hebbian network. Finally, the optimal action sequence \tilde{a}^* that minimizes the distance between the state sequence \tilde{s} and the goal state s^* to be attained is selected and executed by the agent.

In a series of pioneering work using Predictive Coding, A. Ororbia *et al.* [12, 52] have proposed a bio-inspired robot control framework termed Active Predictive Coding (ActPC) and have successfully applied the proposed technique on a range of different robot control tasks from both the *Mujoco* control suite (*Reacher-v4*, *Half-Cheetah-v2*, *Hopper-v2*, *Walker-v2* and *Swimmer-v2*), and from the challenging *Robosuite* environment (*Block Lift* and *Can Place* tasks). By experimentally comparing their proposed system to a range of conventional reinforcement learning techniques, the authors show that their proposed ActPC system is either on-par or either outperforms the use of conventional reinforcement learning techniques while being bio-plausible and only resorting to local Hebbian learning rules. Different to the Hebbian Active Inference system described earlier [51], their system architecture follows an *actor-critic* approach where a first Predictive Coding network plays the role of *actor*, proposing the next action given the current observation. Then, a second Predictive Coding network plays the role of *critic*, estimating the expected reward based on the observation and the action proposed by the *actor* network. These two networks are complemented with additional Hebbian learning mechanisms for encouraging environment exploration and generating goal-seeking behavior, in order to balance the exploration-exploitation trade off (i.e., how much the agent is learning new behavior and how much the agent is seeking to reach the goal).

4.4. Continual Simultaneous Localization and Mapping via Hebbian learning

Simultaneous Localization and Mapping (SLAM) is a fundamental task for robot navigation, where a robot must jointly construct a map of the environment in which it is navigating while localizing itself into that map [53]. Learning-based SLAM systems are usually built by first training a neural network model to perform the visual encoding of

sensory data (e.g., RGB camera) into feature descriptors. Then, these feature descriptors are used to detect new robot poses and locations already visited by the robot in order to perform *loop closure* detection. Once a loop closure is detected, the map that is being built by the SLAM algorithm can be refined and the robot position corrected. But these learning-based SLAM approaches rely on pre-trained neural network using training sets acquired beforehand. This comes in contrast with the fact that these SLAM systems must work in new environments, often *not* captured by static training set acquired *a priori*.

For this reason, there has recently been a growing interest in using continual learning with SLAM, for learning the neural network encoder on the fly, as the robot is exploring unseen environments [38]. Following this line of research, A. Safa *et al.* [13] have proposed a continual learning SLAM system using a Sparse Coding-based Hebbian learning network. The key innovation of their proposed system lies in the use of a Hebbian network similar to the network shown in Fig. 2 in order to encode the input RGB image into lower-dimensional feature descriptors (of dimension 64). Then, the authors in [13] feed the obtained descriptor into a RatSLAM back-end [53] which gradually builds a map and localize the agent by using the feature descriptors for loop closure detection, correcting the raw odometry provided by an Inertial Measurement Unit (IMU). Crucially, the authors in [13] provide a *surprise-modulated* Hebbian learning rule for reaching better continual-learning performance. Each time a new camera observation o_k is received, the following *surprise factor* is computed:

$$\mathcal{S} = \|\Phi_{k-1}c_k - o_k\|_2^2 - \|\Phi_{k-1}c_{k-1} - o_{k-1}\|_2^2 \quad (24)$$

If the surprise factor in (24) is positive ($\mathcal{S} > 0$), the input observation o_k at the current time step k is used to perform a Hebbian learning step (Eq. 7 and 8), refining the dictionary (or weight matrix) Φ . Else, no learning is executed for that time step.

Intuitively, this technique enables the filtration of incoming data points that are deemed redundant and already learned by the system. Indeed, if the incoming input observation o_k leads to a re-projection error $\|\Phi_{k-1}c_k - o_k\|_2^2$ that is *larger* than the previous re-projection error $\|\Phi_{k-1}c_{k-1} - o_{k-1}\|_2^2$ at time step $k-1$, this indicate that the incoming observation has not yet been well captured by the network weights Φ and hence must be incorporated into the learning process. On the other hand, if $\mathcal{S} < 0$, this means that the incoming observation is probably already well captured by the Hebbian network. Hence, integrating that observation point may lead to the over-fitting of the network on the local data stream and lead to catastrophic forgetting (see Section 3.3).

The authors in [13] experimentally validate their findings using a drone flying inside a warehouse environment, by showing that the proposed surprise-driven Hebbian learning method (24) leads to a lower replay error compared to the use of standard Hebbian learning. Furthermore, the authors also show that the proposed surprise-driven method achieves better network stability compared to the use of standard Hebbian learning, where it was observed that the learning process could diverge in some particular network settings.

Table 1. Works in the field of continual Hebbian learning covered in Section 4.

Reference	CL Application	Architecture
A. Ororbia [10]	Task- and Class-Incremental	Predictive Coding
L. Annabi <i>et al.</i> [11]	Sequence Modelling	Predictive Coding
A. Ororbia <i>et al.</i> [63]	Sequence Modelling	Predictive Coding
A. Safa <i>et al.</i> [51]	Agent Control	Sparse Coding
A. Ororbia <i>et al.</i> [12]	Agent Control	Predictive Coding
A. Ororbia <i>et al.</i> [52]	Agent Control	Predictive Coding
A. Safa <i>et al.</i> [13]	SLAM	Sparse Coding

Finally, the authors use the proposed continual Hebbian learning setup to demonstrate a complete SLAM system using a drone flying between storage racks and open areas. Experimental results show that the proposed system reaches either on-par or either better performance compared to state-of-the-art SLAM systems, while not requiring any offline network pre-training nor any explicit training data acquisition step. Hence, the work in [13] opens the door to the application of Continual Hebbian learning for performing SLAM in unknown environment not captured by datasets available beforehand.

Table 1 briefly summarizes the works in the field of continual Hebbian learning that have been covered in this Section. In the next Section, an overview of continual-learning works using SNNs with STDP learning will be presented.

5. Continual learning in SNNs with STDP

5.1. Task-incremental Continual Learning with SNNs and STDP

In a recent paper, D. I. Antonov *et al.* [14] have proposed a first study on how to achieve continual learning with SNNs trained with local STDP rules. A key aspect of their work is that they develop a continual-learning method for determining the importance of each weight in their network using *stochastic Langevin dynamics*, in contrast to the use of gradient-based regularization techniques such as EWC, SI and so on [43, 44], which lack bio-plausibility due to their non-local nature. The network used in their experiments uses two local learning rules: conventional unsupervised STDP and the *supervised* reward-modulated STDP which potentiate the weights when a reward is received (e.g., if the network made the correct prediction) and vice versa in the opposite case. The network architecture consists of three convolutional layers followed by pooling layers, using Integrate and Fire (IF) spiking neurons [64]. The last layer is used to perform classification where the neuron with the earliest spike time denotes the inference decision.

In order to achieve continual learning, the authors in [14] experiment with a number of methods such as; *i)* using winner-take-all (WTA) competition mechanisms between the neurons of each layer (following indications in literature about the importance of

sparsity for continual learning [65]); *ii)* using pseudo-rehearsal where noise patterns are fed to the network and the output is recorded for being used as a pseudo-rehearsal technique for when a new task must be learned; *iii)* using regularization methods by applying noise during training, using dropout [54], freezing the large weights during learning, and using their proposed *Langevin dynamics* technique.

The proposed Langevin dynamics method for continual learning seeks to determine the domain in which each weight w_i can be changed while not affecting the accuracy of the system. By determining this domain D_i , each weight w_i can then be constrained to stay within its particular domain D_i when a new task is being learned, reducing catastrophic forgetting on the previous task. To find such multi-dimensional region of permitted weight values, Langevin dynamics is introduced by applying noise to the weights. During their random exploration of the weight space, when the weights enter a region where the network accuracy is dropping, the R-STDP mechanism then acts as feedback mechanism bringing the weights values back into the admitted region. After enough time, the Brownian motion of the weights will explore the permissible region, which is used to constraint the weights during the learning of new tasks.

Then, the authors in [14] perform a series of experiments in a task-incremental setting using the MNIST and the Extended MNIST (EMNIST) dataset (containing both letters and digits). The task-incremental learning setup is conducted as follows: first, the network is trained using MNIST and then, assuming the MNIST data is not available anymore, the network is then trained using the EMNIST data as the second task. The authors compare the various continual learning techniques explored in their work in terms of global accuracy on both tasks. It is shown that their proposed approaches using Langevin dynamics is highly promising and came close to the performance achieved using *self-reminder* techniques (replay using a subset of the previous task) while being *less memory intensive* than self-reminder techniques and hence, better suited for continual learning application in emerging neuromorphic edge devices. Indeed, Langevin dynamics and related Markov Chain Monte Carlo (MCMC) methods have been found as well-suited for implementation in neuromorphic edge devices though the innovative use of memristor technology, by exploiting the inherent stochasticity of these emerging hardware devices [69].

In another work by A. Ororbia [34], the author proposes a continual learning SNN architecture based on Predictive Coding using an STDP-like Hebbian local learning rule. The framework, termed *spiking neural coding*, can be seen a spiking implementation of the Predictive Coding network architecture proposed by the same authors in [10] and discussed in Section 4.1 (although their SNN implementation does not feature the *task selector* network used in [10]). Their SNN forms a generative model which receives as input tuples $\{X, y\}$ containing the data points X and associated one-hot encoded label y , converted to spike trains using Poisson coding. Then, at test time, only the test data X_{test} is presented to the network which can then estimate the associated label \hat{y} by re-projection of the neural activity into the input space ($o \approx \Phi c$ in (9)).

Then, extensive experimental demonstrations are provided to benchmark the

proposed spiking neural coding architecture against a number of state-of-the-art deep learning techniques used for alleviating catastrophic forgetting (such as EWC, SI and so on [43, 44]). Experiments are conducted using datasets such as MNIST, Fashion MNIST, Stanford Optical Character Recognition and Caltech 101 Silhouettes [55, 56, 57]. After demonstrating the usefulness of their proposed algorithm under the conventional training procedure (i.e., not in a continual learning manner), the author in [34] provide a study of the memory retention ability of their proposed SNN *under the continual learning setting* this time. It is remarkably shown that their proposed spiking neural coding approach obtains highly competitive continual learning performance compared to state-of-the-art methods, outperforming them on both MNIST and Fashion MNIST datasets.

Finally, another SNN architecture for continual learning has been proposed by J. M. Allred *et al.* [15], which differs from the previously covered works through its neuroscience-based use of dopaminergic plasticity modulation for achieving *unsupervised* continual learning. The proposed framework, termed Controlled Forgetting Networks (CFNs), temporarily makes the synaptic weights of some neurons more plastic (i.e., more subject to change), while keeping fixed the weights of other neurons. This allows an isolated and distributed adaptation of the SNN weights through STDP. This selective modulation is achieved by taking close inspiration from biology. In the brain, dopamine acts as a neuromodulator on synaptic plasticity, where dopamine releases are associated with the encountering of novel, surprising data points. This surprise-driven synaptic modulation is achieved by scaling the STDP learning rate of each neuron through a set of additional *dopaminergic* neurons that take as input the spiking activity of neurons in each layer and feed back their output to each neuron in that layer in order to modulate their individual learning rate.

Experimental results are provided using the MNIST dataset and it is shown that the proposed method successfully achieves continual learning, retaining an accuracy larger than 90% after learning each digit class in MNIST one by one. In addition, the authors conduct ablation studies to show the usefulness of their various design choices and their newly-proposed dopaminergic neurons, where not using these aforementioned techniques lead to a severe decrease in accuracy during the continual learning process.

Hence, the works discussed above show the usefulness of SNNs in continual learning settings and strongly motivates their exploration in other continual learning contexts, beyond task-incremental learning.

5.2. Continual SNN-STDP learning for Robotics applications

A number of SNN-based works have been studied in literature for performing Simultaneous Localization and Mapping (SLAM) [58, 59], but these work usually rely on the offline training of the SNN visual encoder networks, instead of using continual learning, which is the focus of this survey.

In order to extend SNN-based robotics systems to the continual learning case, a

number of recent works have been exploring the use of SNN equipped with *continual* STDP learning in robotics applications, with a focus on solving various drone navigation tasks such as Simultaneous Localization and Mapping (SLAM) and people detection from drones.

A first-of-its-kind SLAM system has been proposed by A. Safa *et al.* [9] which can continuously learn to encode sensory data on the fly into feature descriptors (using an event-based camera and a radar as input sensors to their SNN-STDP network). The authors in [9] set up an SNN-STDP architecture which essentially acts as a spiking substrate of computation for Sparse Coding (introduced in Section 3.1.1). Upon receiving the spiking sensory data, the SNN-STDP continuously refines its weights on the fly as the drone is flying in an indoor warehouse-like environment and jointly produces a spiking output sparse code. These output spike trains are averaged over a time window $\Delta T = 50$ ms to obtain continuous-valued feature descriptors. Finally, the produced feature descriptors are fed to a RatSLAM back-end [53] for performing loop closure detection and map correction.

Then, the authors in [9] benchmark their proposed continual SNN-STDP SLAM algorithm against a number of CNN-based and feature-based SLAM approaches by mounting an indoor Ultra Wide Band (UWB) positioning beacon on the drone for acquiring ground truth localization and mapping data. It is shown that the proposed approach either reaches on-par or either outperforms the competing SLAM methods while being extremely robust to lighting variation, due to its use of a radar sensor coupled with an event-based camera (which typically performs well in low-light due to its high dynamic range [60]).

A remarkable aspect of their system is the fact that the SNN is never pre-trained on an offline acquired dataset. Rather, their SNN-STDP network is initialized randomly and continuously refines its weights on the fly as the drone explores new unknown environments. Interestingly, the authors show in [61] that the *continual learning nature* of their proposed SNN-STDP system brings novel value from an algorithmic point of view, by outperforming the same SNN-STDP architecture but *trained offline* using a pre-recorded dataset (see Fig. 8).

This first demonstration of a continual learning SLAM system using SNN-STDP opens the door for the conception of novel robots that can learn to navigate in unseen environment without resorting to training sets acquired offline.

In another work by A. Safa *et al.* [7], an attention-based people detection system has been proposed for detecting potential human subjects walking in front of a flying drone in order to provide collision avoidance. Their proposed system makes use of an innovative continual learning SNN-STDP network built following a fully-convolutional network architecture. Indeed, a fully convolutional architecture is utilized in order to provide *translational invariance* (i.e., detecting people in an invariant manner regardless of their position on the image plane). The continual learning process of their system works as follows. First, a drone embarking an event-based camera (which provides spiking image data) is made to fly in an indoor environment where human subjects are

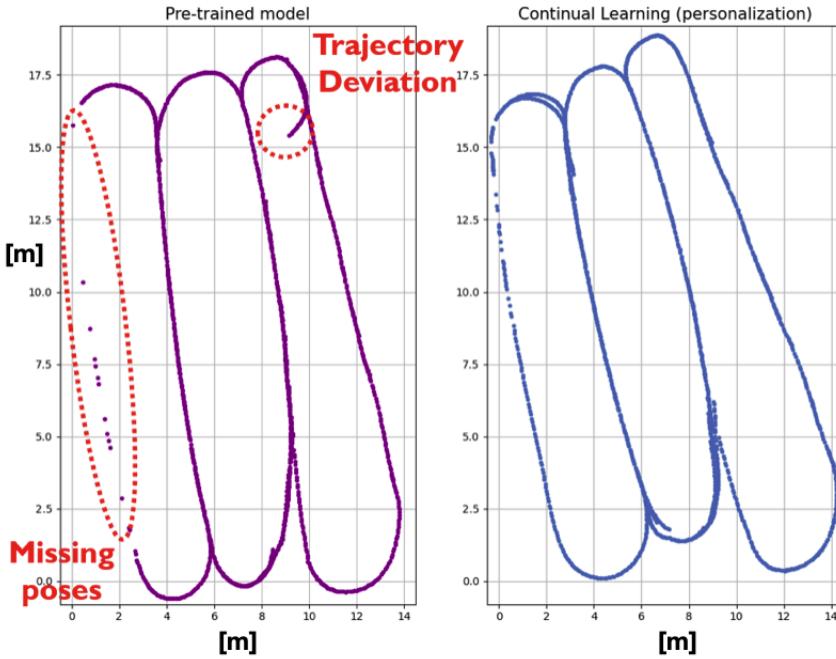


Figure 8. Pre-training vs. Continual Learning for the continual learning SNN-STDP SLAM system proposed in [9, 61]. The system trained offline suffers from a significant number of missing poses and trajectory deviations. The axis on the graph have dimensions in meters [m].

walking in front of the drone. During this flight, the SNN-STDP system is continuously learning to output an attention map where high pixel values denote the presence of a human subject (see Fig. 9). After a ~ 2 minute learning sequence, the SNN weights are fixed and the complete system is assessed on other flight sequences in order to measure the detection performance of the proposed people detection architecture.

During the continual learning phase, the SNN learning is done through a semi-supervised STDP-driven setup, where a supervised STDP contribution modifies the SNN weights using the human location labels, and an unsupervised STDP contribution is used as a regularization method providing better detection performance by reducing the amount of false alarms in the produced attention maps. Remarkably, the authors in [7] show that their proposed continual learning SNN-STDP system outperforms the use of a deep CNN *trained offline* by +19% in terms of human detection performance using the event-based camera data. It is argued that this superior performance is linked to the recurrent nature of the proposed SNN-STDP system which better integrates the timing information embedded in the event-based camera data stream.

Together with the SLAM system proposed in [9], the SNN-STDP-based people detection system presented in [7] constitute what is, to the best of our knowledge, some of the most advanced real-world applications of continual SNN-STDP learning demonstrated in the robotics context.

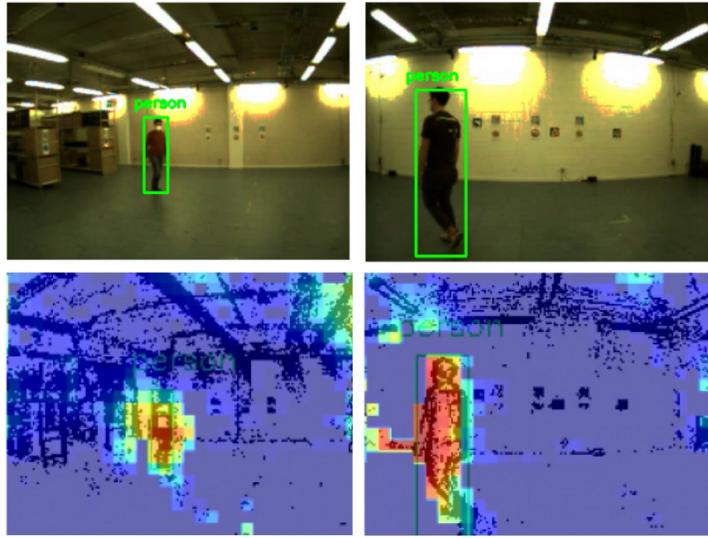


Figure 9. Attention maps from the DVS data, inferred by the continual-learning SNN-STDP system in [7]. The label bounding boxes are also shown.

5.3. Continual SNN-STDP implementations in Integrated Circuits

An important direction for continual SNN-STDP systems lies in the design of novel Integrated Circuits (ICs) or chips that can perform continual learning at the extreme edge with reduced energy and area utilization. Indeed, building such novel ICs will enable the conception of a novel range of IoT and robotics devices that can be personalized on the fly to their particular environment.

Within this context, I. Munoz-Martin, S. Bianchi *et al.* [16, 62] have studied in a number of works the conception of a continual learning system using unsupervised STDP learning implemented in hardware using CMOS logic and phase change memory (PCM) circuits for implementing the SNN synapses. The authors in [16, 62] build a mixed supervised-unsupervised network comprising a supervised convolutional network feeding into an unsupervised STDP network with Winner-Take-All behavior serving as the final classification layer.

The continual learning scenario demonstrated in [16, 62] works as follows. The network is first trained using the seven first classes of the MNIST dataset in an offline training manner. Then, the remaining three last classes are learned by the sole STDP-driven adaptation of the output SNN layer (without access to the data of the first seven classes), while still retaining some knowledge of the past seven classes thanks to the Winner-Take-All mechanism. Using this continual learning setup, the authors in [16, 62] reach an overall accuracy of 93%. The same experiment is repeated with the CIFAR10 dataset, resulting in an overall accuracy of 82% after continual learning. In essence, the authors in [16, 62] achieve a task-based continual learning setup comprised of two separate tasks to be learned.

An innovative aspect of the work in [16, 62] lies in the use of Phase Change Memory (PCM) material to store the synaptic weights of the SNN in the proposed

Table 2. Works in the field of continual STDP learning covered in Section 5.

Reference	CL Application	Architecture
D. I. Antonov <i>et al.</i> [14]	Task-Incremental	Multi-layer Convolutional SNN-STDP
A. Ororbia [34]	Task-Incremental	Predictive Coding via multi-layer SNN-STDP
J. M. Allred <i>et al.</i> [15]	Task-Incremental	Excitatory-Inhibitory with Dopaminergic feedback
A. Safa <i>et al.</i> [9]	SLAM	Recurrent SNN-STDP
A. Safa <i>et al.</i> [7]	People Detection	Convolutional SNN-STDP
I. Munoz-Martin, S. Bianchi <i>et al.</i> [16, 62]	Task-Incremental	CNN feature map - SNN-STDP

circuit implementation. For this hardware perspective, the work in [16, 62] significantly departs from conventional *digital* SNN-STDP processor design by exploring the use of highly novel and exploratory circuit elements such as PCMs. PCMs are a class of resistive memory that can store the synapse value as the value of their electrical resistance. A key aspect of PCM devices is that their electrical resistance value can be altered depending on the current and signals applied on the PCM terminals. This makes them particularly well-suited for local learning rules such as STDP which only depend on pre-synaptic and post-synaptic neural activity. Regarding energy consumption, their proposed system requires an average energy consumption of $\sim 0.1 \mu\text{J}$, which makes the proposed PCM-based SNN-STDP solution well-suited for extreme edge AI application. It is important to precise that other types of innovative nanodevices beyond PCMs are being currently studied by many teams for implementing synaptic weights, such as memristors [69], ferroelectric memories [70] and magnetoresistive memories [71].

The work proposed in [16, 62] by I. Munoz-Martin, S. Bianchi *et al.* constitutes a major stepping stone for the conception of energy-efficient edge AI devices via innovative circuit technologies that can continuously learn and personalize their underlying SNN model to new tasks and to changes in their working environment.

Table 2 briefly summarizes the works in the field of continual STDP learning that have been covered in this Section.

6. Discussion and open future research questions

The emerging number of works in neuromorphic continual learning covered in the Sections 4 and 5 of this paper have clearly demonstrated the usefulness of bio-plausible neuromorphic architectures based on Sparse and Predictive Coding, as well as on Spiking Neural Networks with STDP learning for a wide range of different continual-learning scenarios and applications. These different applications ranged from task-incremental and class-incremental learning to drone navigation use cases such as Simultaneous Localization and Mapping (SLAM) and people detection on drones. In this Section, we aim at providing a discussion on the future perspectives for the field of neuromorphic continual learning so as to identify both the open research questions and the future

application domains where we expect continual learning systems based on SNNs and other hardware-efficient neuromorphic technologies to bring great value.

6.1. Neuromorphic task-incremental learning: open research questions

In light of the various task-incremental works surveyed in this paper [10, 14, 34, 15], using both non-spiking Hebbian networks and SNN-STDP architectures, we can now formulate a number of open research questions for further pushing the boundaries of research in neuromorphic task-incremental learning systems:

- A number of works based on Sparse and Predictive Coding using both non-spiking Hebbian learning and spiking SNN-STDP have been proposed, but none of them is built using a *convolutional* architecture. Indeed, in the fully-connected cases, the input data and the labels are all fed as inputs to the predictive coding setups during learning. Then, during inference, only the data is fed as input and the labels are inferred by re-projection of the neural activity back into the input space. Even though this technique works for the fully-connected case, it is not fully clear if the same process would be effective in the convolutional case, since feeding both 2D images and 1D label vectors during learning could lead to a degraded performance, due to the signal dimensionality imbalance between the 2D image and 1D label case.
- Another interesting direction of research would be to explore how regularization-based continual learning techniques found in deep learning literature [8] could be fused with neuromorphic continual learning techniques in order to combine the advantages of both fields. Hence, it would be interesting to study how to apply e.g., Elastic Weight Consolidation [43] to the Predictive Coding Hebbian and SNN-STDP learning systems proposed in [10, 14, 34, 15].
- Finally, a number of works have pointed to possible links between both Hebbian and STDP-based learning, and the theory of Bayesian learning [51, 72]. indeed, Bayesian learning theory provides a principled way to study how continual learning can be achieved since the *iterative Bayesian update* is insensitive to catastrophic forgetting. Iterative Bayesian update is performed by inferring a Posterior distribution $P(w|D)$ over the neural network weights w given the data D , using the well-known Bayes' rule $P(w|D) = \frac{L(D|w)P(w)}{P(D)}$ (with $L(D|w)$ the likelihood distribution and $P(w)$ the prior) [66]. In fact, a number of regularization-based continual learning systems used in deep learning (such as the Elastic Weight Consolidation technique [43]) can be directly linked to the iterative Bayes update, under a number of approximations and simplifications [43]. As future research, the same could be done for the neuromorphic case, by studying how Bayesian learning can be formulated in Hebbian learning and SNN-STDP systems.

6.2. Continual learning in neuromorphic robotics systems: open research questions

Following the various robotics-related works covered in this paper [13, 9, 7], we can formulate the following open research questions for the field of neuromorphic continual learning research in robotics:

- An emerging number of work exploring the use of both SC-based Hebbian learning [13] and SNN-STDP learning [9] for Simultaneous Localization and Mapping have been proposed. Thanks to their continual learning nature, the neural networks used in these works did not need to be trained offline but were rather learned online, as the drone was exploring the unknown environment. Even though useful, the demonstrations in [13, 9, 7] have been provided in a single type of environment (an indoor warehouse). Hence, it might be interesting to investigate how these continual learning neuromorphic SLAM system behave when navigating from one environment to a totally different one (e.g., going from indoor to outdoor). It is expected that in this case, multiple Hebbian or SNN-STDP networks will be needed with each one dedicated to a particular environment. Then, a higher-order system detecting when the robot is passing from one environment to another can be used to switch between the different Hebbian and SNN-STDP ensembles (similar to the task selector network in [10]).
- The work in [7] has studied the continual learning of people detection from an event-based camera mounted on a drone. In [7], continual learning is done by using a stream of provided labels indicating where walking people are present on the output attention maps. In order to increase the usability of neuromorphic continual learning perception systems, it might be interesting to study how the use of precise labels in [7] could be dropped and replaced with the use of e.g., binary reward signals indicating if the drone is coming too close to a human subject or not. Indeed, it is not expected that precise labels are available during the continual learning process for practical use cases. Hence, studying how binary alarm-type signals could be used instead of precise labels will help making continual learning neuromorphic perception systems such as [7] better suited for practical use cases.
- In light of the emerging number of work about the use of Hebbian learning for agent control [51, 12, 52], more research is needed for bridging the gap between continual learning neuromorphic systems and classical reinforcement learning techniques. Works such as [51] have shown the benefits of continual Hebbian learning in ANNs over conventional reinforcement learning setups in terms of a faster learning convergence and a reduced need for memory-expensive replay buffers. On the other hand, these demonstrations are still limited to rather simple reinforcement learning environments (such as the mountain car environment in [51]). Therefore, more research is needed for extending the applicability of neuromorphic agent control systems based on Hebbian learning rules to more complex problems.

6.3. Design of novel neuromorphic hardware for continual learning

A major promise of neuromorphic computing hardware resides in its ultra-low-power aspect, as opposed to the more resource-expensive deep learning accelerator hardware [68]. The power-efficient aspect of neuromorphic system is even more crucial when it comes to providing *learning* capability in extreme edge AI devices. Indeed, conventional deep learning systems rely on the *back-propagation of error* algorithm during training which is known to be significantly more compute-expensive compared to the local learning rules such as Hebbian and STDP learning [17, 5].

Since providing learning at the edge is believed to be a key enabler of the next generation of edge AI devices that can continuously personalize their models to their specific environments, a fast-growing body of work for the design of SNN hardware equipped with energy-efficient STDP learning has emerged. Still, most of the SNN-STDP hardware designs that have been proposed in literature are demonstrated in a *continual learning* scenario but rather work in conventional offline training settings (with the exception of [16, 62] among others).

Hence, it is important to put a greater focus on the design of SNN-STDP hardware that can operate in continual learning settings. In order to achieve this aim, novel research directions could be adopted, such as:

- Closely collaborating with the neuromorphic algorithm research community in order to implement successful continual learning SNN-STDP designs into dedicated integrated circuits, following an algorithm-hardware co-design approach.
- Studying how regularization-based continual learning methods can be simplified and implemented in dedicated hardware accelerators, since this class of method provide a good balance between performance and compute complexity in comparison to e.g., memory replay methods [67].
- Exploring how the currently-existing SNN-STDP hardware designs can still be utilized in a continual learning settings, and under which conditions with which impact on the performance.

7. Conclusion

This paper has provided an overview of the rapidly-evolving field of neuromorphic continual learning using Sparse and Predictive Coding architectures, with Hebbian learning and its STDP variant in the spiking domain. First, this paper has motivated and described the background theory needed for the study of bio-plausible neural networks based on Sparse and Predictive Coding, learning via Hebbian and STDP rules. This paper has also provided background on continual learning by introducing the various methods used in literature to learn in non-stationary data streams. Then, after introducing the background theory, this paper has reviewed a number of recent work in the field of neuromorphic continual learning. The works surveyed in this paper

have covered a wide range of different applications, from task-incremental and class-incremental learning to robotics, Simultaneous Localization and Mapping, and agent control. Finally, this paper has provided a future perspective on the open research questions for the field of neuromorphic continual learning. It is hoped that this survey paper will be useful for the wider neuromorphic research community and will inspire future research in bio-plausible continual learning.

8. References

- [1] Schuman, C.D., Kulkarni, S.R., Parsa, M. et al. Opportunities for neuromorphic computing algorithms and applications. *Nat Comput Sci* 2, 10–19 (2022). <https://doi.org/10.1038/s43588-021-00184-y>
- [2] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, James S. Plank. (2017). "A Survey of Neuromorphic Computing and Neural Networks in Hardware."
- [3] J. D. Nunes, M. Carvalho, D. Carneiro and J. S. Cardoso, "Spiking Neural Networks: A Survey," in *IEEE Access*, vol. 10, pp. 60738-60764, 2022, doi: 10.1109/ACCESS.2022.3179968.
- [4] Ben Abdallah, A., N. Dang, K. (2022). Survey of Neuromorphic Systems. In: *Neuromorphic Computing Principles and Organization*. Springer, Cham. https://doi.org/10.1007/978-3-030-92525-3_9
- [5] Lyes Khacef, Philipp Klein, Matteo Cartiglia, Arianna Rubino, Giacomo Indiveri, Elisabetta Chicca (2023). Spike-based local synaptic plasticity: a survey of computational models and neuromorphic circuits. *Neuromorphic Computing and Engineering* , 3(4), 042001.
- [6] Mishra, R., Suri, M. (2023). "A survey and perspective on neuromorphic continual learning systems." *Frontiers in Neuroscience*, 17.
- [7] A. Safa, I. Ocket, A. Bourdoux, H. Sahli, F. Catthoor and G. G. E. Gielen, "STDP-Driven Development of Attention-Based People Detection in Spiking Neural Networks," in *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 1, pp. 380-387, Feb. 2024, doi: 10.1109/TCDS.2022.3210278.
- [8] M. De Lange et al., "A Continual Learning Survey: Defying Forgetting in Classification Tasks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3366-3385, 1 July 2022, doi: 10.1109/TPAMI.2021.3057446.
- [9] A. Safa et al., "Fusing Event-based Camera and Radar for SLAM Using Spiking Neural Networks with Continual STDP Learning," 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023, pp. 2782-2788, doi: 10.1109/ICRA48891.2023.10160681.
- [10] Alex Ororbia, Ankur Mali, C. Lee Giles, Daniel Kifer (2022). Lifelong Neural Predictive Coding: Learning Cumulatively Online without Forgetting. In *Advances in Neural Information Processing Systems*.
- [11] Annabi, L., Pitti, A., Quoy, M. (2022). "Continual Sequence Modeling With Predictive Coding." *Frontiers in Neurorobotics*, 16.
- [12] A. Ororbia and A. Mali, "Active Predictive Coding: Brain-Inspired Reinforcement Learning for Sparse Reward Robotic Control Problems," 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023, pp. 3015-3021, doi: 10.1109/ICRA48891.2023.10160530.
- [13] A. Safa et al., "Learning to Encode Vision on the Fly in Unknown Environments: A Continual Learning SLAM Approach for Drones," 2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Sevilla, Spain, 2022, pp. 373-378, doi: 10.1109/SSRR56537.2022.10018713.

- [14] D.I. Antonov, K.V. Sviatov, S. Sukhov (2022). Continuous learning of spiking networks trained with local rules. *Neural Networks*, 155, 512-522.
- [15] Allred, J., Roy, K. (2020). Controlled Forgetting: Targeted Stimulation and Dopaminergic Plasticity Modulation for Unsupervised Lifelong Learning in Spiking Neural Networks. *Frontiers in Neuroscience*, 14.
- [16] I. Muñoz-Martín, S. Bianchi, G. Pedretti, O. Melnic, S. Ambrogio and D. Ielmini, "Unsupervised Learning to Overcome Catastrophic Forgetting in Neural Networks," in IEEE Journal on Exploratory Solid-State Computational Devices and Circuits, vol. 5, no. 1, pp. 58-66, June 2019, doi: 10.1109/JXCD.2019.2911135.
- [17] A. Safa, J. Van Assche, M. D. Alea, F. Catthoor and G. G. E. Gielen, "Neuromorphic Near-Sensor Computing: From Event-Based Sensing to Edge Learning," in IEEE Micro, vol. 42, no. 6, pp. 88-95, 1 Nov.-Dec. 2022, doi: 10.1109/MM.2022.3195634.
- [18] Friston, K., Kiebel, S. (2009). "Predictive coding under the free-energy principle." *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 364, 1211-21.
- [19] Bruno A. Olshausen, David J. Field (1997). "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vision Research*, 37(23), 3311-3325.
- [20] Friston, K. Does predictive coding have a future?. *Nat Neurosci* 21, 1019–1021 (2018). <https://doi.org/10.1038/s41593-018-0200-7>
- [21] Umais Zahid, Qinghai Guo, Zafeirios Fountas.(2023). "Predictive Coding as a Neuromorphic Alternative to Backpropagation: A Critical Evaluation."
- [22] Jiang Hao, Thomas G Oertner (2012). Depolarization gates spine calcium transients and spike-timing-dependent potentiation. *Current Opinion in Neurobiology*, 22(3), 509-515.
- [23] Safa, A., Ocket, I., Bourdoux, A., Sahli, H., Catthoor, F., Gielen, G. (2022). "Event Camera Data Classification Using Spiking Networks with Spike-Timing-Dependent Plasticity." In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8).
- [24] Yuchen Liang, Chaitanya Ryali, Benjamin Hoover, Leopold Grinberg, Saket Navlakha, Mohammed J Zaki, Dmitry Krotov (2021). "Can a Fruit Fly Learn Word Embeddings?." In International Conference on Learning Representations.
- [25] Ablin, P., Moreau, T., Massias, M., Gramfort, A. (2019). "Learning step sizes for unfolded sparse coding." In Advances in Neural Information Processing Systems. Curran Associates, Inc.
- [26] Tsung-Han Lin, Ping Tak Peter Tang (2019). "Sparse Dictionary Learning by Dynamical Neural Networks." In International Conference on Learning Representations.
- [27] Millidge, B., Seth, A., Buckley, C. (2021). "Predictive Coding: a Theoretical and Experimental Review." arXiv preprint arXiv:2107.12979.
- [28] J. K. Eshraghian et al., "Training Spiking Neural Networks Using Lessons From Deep Learning," in Proceedings of the IEEE, vol. 111, no. 9, pp. 1016-1054, Sept. 2023, doi: 10.1109/JPROC.2023.3308088
- [29] Eliasmith, C., Anderson, C. (2003). "Neural Engineering: Computation, Representation and Dynamics in Neurobiological Systems." MIT press.
- [30] Frenkel, C., Lefebvre, M., Bol, D. (2021). "Learning Without Feedback: Fixed Random Learning Signals Allow for Feedforward Training of Deep Neural Networks." *Frontiers in Neuroscience*, 15.
- [31] Bi, G., Poo, M. (1998). "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type." *J. Neurosci.*, 18(24), 10464–10472.
- [32] Montangie, L., Miehl, C., Gjorgjieva, J. (2020). "Autonomous emergence of connectivity assemblies via spike triplet interactions." *PLoS Comput. Biol.*, 16(5), e1007835.
- [33] Ali Safa, Ilja Ocket, André Bourdoux, Hichem Sahli, Francky Catthoor, Georges Gielen. (2022). "A New Look at Spike-Timing-Dependent Plasticity Networks for Spatio-Temporal Feature Learning."
- [34] Alexander Ororbia (2023). Spiking neural predictive coding for continually learning from data streams. *Neurocomputing*, 544, 126292.

- [35] Legenstein R, Pecevski D, Maass W (2008) A Learning Theory for Reward-Modulated Spike-Timing-Dependent Plasticity with Application to Biofeedback. *PLoS Comput Biol* 4(10): e1000180. <https://doi.org/10.1371/journal.pcbi.1000180>
- [36] M. De Lange and T. Tuytelaars, "Continual Prototype Evolution: Learning Online from Non-Stationary Data Streams," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 8230-8239, doi: 10.1109/ICCV48922.2021.00814.
- [37] Bengio Y, Mesnard T, Fischer A, Zhang S, Wu Y. STDP-Compatible Approximation of Backpropagation in an Energy-Based Model. *Neural Comput.* 2017 Mar;29(3):555-577.
- [38] N. Vödisch, D. Cattaneo, W. Burgard and A. Valada, "CoVIO: Online Continual Learning for Visual-Inertial Odometry," 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Vancouver, BC, Canada, 2023, pp. 2464-2473, doi: 10.1109/CVPRW59228.2023.00245.
- [39] R. Aljundi, K. Kelchtermans and T. Tuytelaars, "Task-Free Continual Learning," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 11246-11255, doi: 10.1109/CVPR.2019.01151.
- [40] Rahaf Aljundi, Marcus Rohrbach, Tinne Tuytelaars (2019). "Selfless Sequential Learning." In International Conference on Learning Representations.
- [41] Qingfeng Lan, A. Rupam Mahmood. (2024). "Elephant Neural Networks: Born to Be a Continual Learner."
- [42] Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T. (2018). Memory Aware Synapses: Learning What (not) to Forget. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science(), vol 11207. Springer, Cham.
- [43] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, Raia Hadsell (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13), 3521-3526.
- [44] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17). JMLR.org, 3987–3995.
- [45] P. J. Werbos, "Backpropagation through time: what it does and how to do it," in Proceedings of the IEEE, vol. 78, no. 10, pp. 1550-1560, Oct. 1990, doi: 10.1109/5.58337.
- [46] Dua, D., and Graff, C. (2019). "Uci machine Learning Repository." Irvine, CA: University of California, School of Information and Computer Science. Available online at: <http://archive.ics.uci.edu/ml>.
- [47] Herbert Jaeger. (2014). Conceptors: an easy introduction.
- [48] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," in *Neural Computation*, vol. 1, no. 2, pp. 270-280, June 1989, doi: 10.1162/neco.1989.1.2.270.
- [49] Corentin Tallec, Yann Ollivier (2018). Unbiased Online Recurrent Optimization. In International Conference on Learning Representations.
- [50] Parr, T., Pezzulo, G., Friston, K. (2022). Active Inference: The Free Energy Principle in Mind, Brain, and Behavior. The MIT Press.
- [51] Safa, A. et al. (2024). Active Inference in Hebbian Learning Networks. In: Buckley, C.L., et al. Active Inference. IWAI 2023. Communications in Computer and Information Science, vol 1915. Springer, Cham.
- [52] Ororbia, A.G., Mali, A.: Backprop-free reinforcement learning with active neural generative coding. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 1, pp. 29–37 (2022)
- [53] M. J. Milford, G. F. Wyeth and D. Prasser, "RatSLAM: a hippocampal model for simultaneous

- localization and mapping," IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, New Orleans, LA, USA, 2004, pp. 403-408 Vol.1, doi: 10.1109/ROBOT.2004.1307183.
- [54] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56), 1929–1958.
 - [55] LeCun, Y., Cortes, C. (2010). MNIST handwritten digit database.
 - [56] Han Xiao, Kashif Rasul, Roland Vollgraf (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR*, abs/1708.07747.
 - [57] Li, F.-F., Andreetto, M., Ranzato, M., Perona, P. (2022). Caltech 101 (1.0) [Data set]. CaltechDATA. <https://doi.org/10.22002/D1.20086>
 - [58] S. Hussaini, M. Milford and T. Fischer, "Ensembles of Compact, Region-specific & Regularized Spiking Neural Networks for Scalable Place Recognition," 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023, pp. 4200-4207, doi: 10.1109/ICRA48891.2023.10160749.
 - [59] S. Hussaini, M. Milford and T. Fischer, "Spiking Neural Networks for Visual Place Recognition Via Weighted Neuronal Assignments," in *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4094-4101, April 2022, doi: 10.1109/LRA.2022.3149030.
 - [60] G. Gallego, et al., "Event-Based Vision: A Survey" in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 01, pp. 154-180, 2022. doi: 10.1109/TPAMI.2020.3008413
 - [61] A. Safa, 2023 SUPPLEMENTARY MATERIAL: Fusing Event-based Camera and Radar for SLAM Using Spiking Neural Networks with Continual STDP Learning. OFFLINE TRAINING VS. CONTINUAL LEARNING.
 - [62] S. Bianchi, I. Muñoz-Martin, G. Pedretti, O. Melnic, S. Ambrogio and D. Ielmini, "Energy-efficient continual learning in hybrid supervised-unsupervised neural networks with PCM synapses," 2019 Symposium on VLSI Circuits, Kyoto, Japan, 2019, pp. 1-2, doi: 10.23919/VLSIC.2019.8778001.
 - [63] A. Ororbia, A. Mali, C. L. Giles and D. Kifer, "Continual Learning of Recurrent Neural Networks by Locally Aligning Distributed Representations," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 10, pp. 4267-4278, Oct. 2020, doi: 10.1109/TNNLS.2019.2953622.
 - [64] A. Safa et al., "Improving the Accuracy of Spiking Neural Networks for Radar Gesture Recognition Through Preprocessing," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 6, pp. 2869-2881, June 2023, doi: 10.1109/TNNLS.2021.3109958.
 - [65] H. Jin, G. Kim, C. Ahn and E. Kim, "Growing a Brain with Sparsity-Inducing Generation for Continual Learning," in 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2023 pp. 18915-18924. doi: 10.1109/ICCV51070.2023.01738
 - [66] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine and M. Bennamoun, "Hands-On Bayesian Neural Networks—A Tutorial for Deep Learning Users," in *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 29-48, May 2022, doi: 10.1109/MCI.2022.3155327.
 - [67] Gido M. van de Ven, Nicholas Soures, Dhireesha Kudithipudi. (2024). Continual Learning and Catastrophic Forgetting.
 - [68] C. Frenkel, M. Lefebvre, J. -D. Legat and D. Bol, "A 0.086-mm² 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28-nm CMOS," in *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 1, pp. 145-158, Feb. 2019, doi: 10.1109/TBCAS.2018.2880425.
 - [69] Dalgaty, T., Castellani, N., Turck, C. et al. "In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling." *Nat Electron* 4, 151–161 (2021). <https://doi.org/10.1038/s41928-020-00523-3>
 - [70] Dong Wang, Shenglan Hao, Brahim Dkhil, Bobo Tian, Chungang Duan (2023). "Ferroelectric materials for neuroinspired computing applications." *Fundamental Research*.
 - [71] Lequeux, S., Sampaio, J., Cros, V. et al. "A magnetic synapse: multilevel spin-torque memristor with perpendicular anisotropy." *Sci Rep* 6, 31510 (2016). <https://doi.org/10.1038/srep31510>

- [72] Nessler, B., Pfeiffer, M., Maass, W. (2008). "Hebbian Learning of Bayes Optimal Decisions." In Advances in Neural Information Processing Systems. Curran Associates, Inc..
- [73] Ororbia, A., Kifer, D. The neural coding framework for learning generative models. *Nat Commun* 13, 2064 (2022). <https://doi.org/10.1038/s41467-022-29632-7>
- [74] Averbeck, B., Latham, P. & Pouget, A. Neural correlations, population coding and computation. *Nat Rev Neurosci* 7, 358–366 (2006). <https://doi.org/10.1038/nrn1888>
- [75] Song Y, Lukasiewicz T, Xu Z, Bogacz R. Can the Brain Do Backpropagation? -Exact Implementation of Backpropagation in Predictive Coding Networks. *Adv Neural Inf Process Syst.* 2020;33:22566-22579. PMID: 33840988; PMCID: PMC7610561.
- [76] Paszke, A. et al., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035
- [77] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.